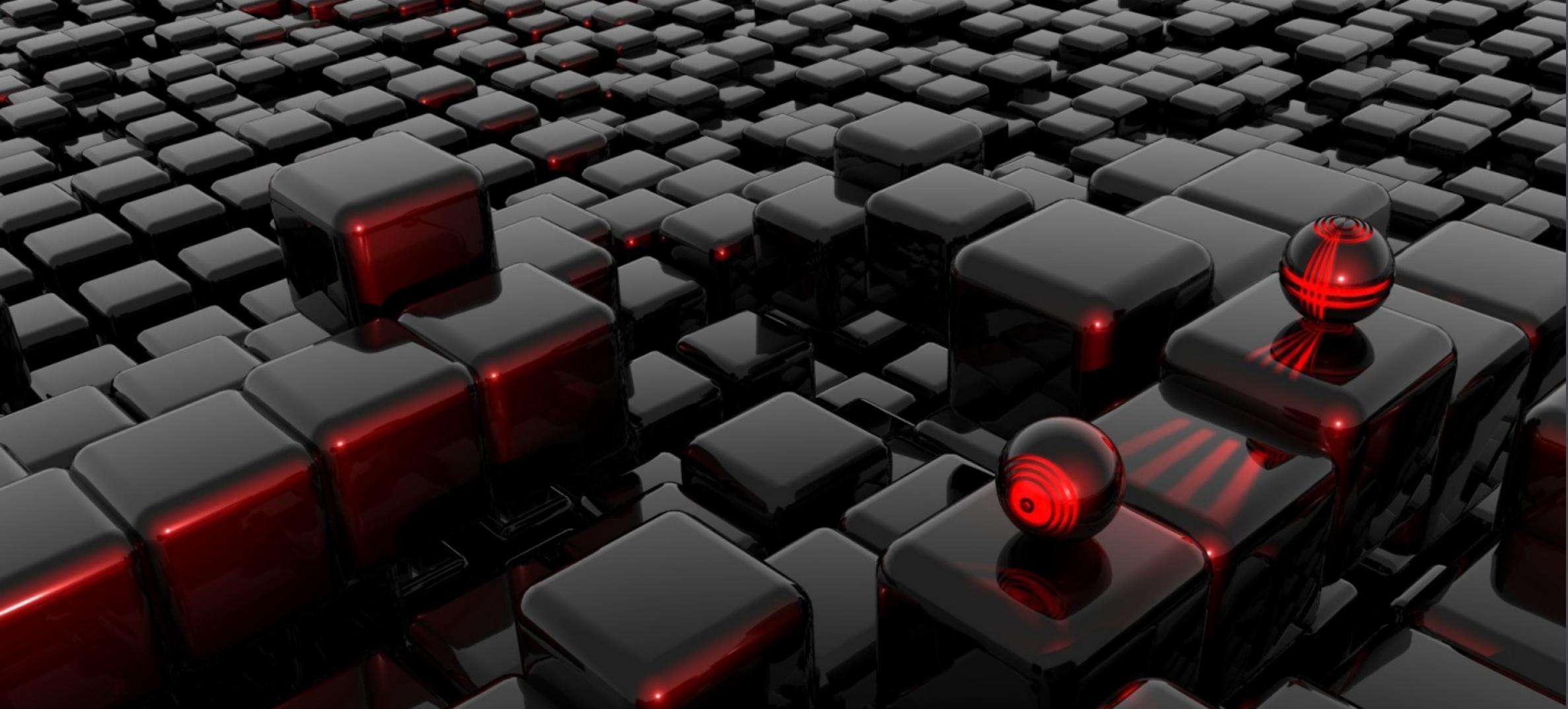




SOFTWARE  
**CIMATEC**





# Native Objects



# #Lista de Objetos e suas funções

- Array.prototype.concat
- Array.prototype.constructor
- Array.prototype.every
- Array.prototype.filter
- Array.prototype.forEach
- Array.prototype.indexOf
- Array.prototype.join
- Array.prototype.lastIndexOf
- Array.prototype.map
- Array.prototype.pop
- Array.prototype.push
- Array.prototype.reduce
- Array.prototype.reduceRight
- Array.prototype.reverse
- Array.prototype.shift
- Array.prototype.slice
- Array.prototype.some
- Array.prototype.sort
- Array.prototype.splice
- Array.prototype.toLocaleString
- Array.prototype.toString
- Array.prototype.unshift
- Boolean.prototype.constructor
- Boolean.prototype.toString
- Boolean.prototype.valueOf
- Date.prototype.constructor
- Date.prototype.getDate
- Date.prototype.getDay
- Date.prototype.getFullYear
- Date.prototype.getHours
- Date.prototype.getMilliseconds
- Date.prototype.getMinutes
- Date.prototype.getMonth
- Date.prototype.getSeconds
- Date.prototype.getTime
- Date.prototype.getTimezoneOffset
- Date.prototype.getUTCDate
- Date.prototype.getUTCDay
- Date.prototype.getUTCFullYear
- Date.prototype.getUTCHours
- Date.prototype.getUTCMilliseconds
- Date.prototype.getUTCMinutes
- Date.prototype.getUTCMonth
- Date.prototype.getUTCSeconds
- Date.prototype.getYear
- Date.prototype.setDate
- Date.prototype.setFullYear
- Date.prototype.setHours
- Date.prototype.setMilliseconds
- Date.prototype.setMinutes
- Date.prototype.setMonth
- Date.prototype.setSeconds
- Date.prototype.setTime
- Date.prototype.setUTCDate
- Date.prototype.setUTCFullYear
- Date.prototype.setUTCHours
- Date.prototype.setUTCMilliseconds
- Date.prototype.setUTCMinutes
- Date.prototype.setUTCMonth
- Date.prototype.setUTCSeconds
- Date.prototype.setYear
- Date.prototype.toDateString
- Date.prototype.toGMTString
- Date.prototype.toISOString
- Date.prototype.toJSON
- Date.prototype.toLocaleDateString
- Date.prototype.toLocaleString
- Date.prototype.toLocaleTimeString
- Date.prototype.toString
- Date.prototype.toTimeString
- Date.prototype.toUTCString
- Date.prototype.valueOf
- Error.prototype.constructor
- Error.prototype.message
- Error.prototype.name
- Error.prototype.toString
- Function.prototype.apply
- Function.prototype.bind
- Function.prototype.call
- Function.prototype.constructor
- Function.prototype.toString
- NativeError.prototype.constructor
- NativeError.prototype.message
- NativeError.prototype.name
- Number.prototype.constructor
- Number.prototype.toExponential
- Number.prototype.toFixed
- Number.prototype.toLocaleString
- Number.prototype.toPrecision
- Number.prototype.toString
- Number.prototype.valueOf
- Object.prototype.constructor
- Object.prototype.hasOwnProperty
- Object.prototype.isPrototypeOf
- Object.prototype.propertyIsEnumerable
- Object.prototype.toLocaleString
- Object.prototype.toString
- RegExp.prototype.constructor
- RegExp.prototype.exec
- RegExp.prototype.test
- RegExp.prototype.toString
- String.prototype.charAt
- String.prototype.charCodeAt
- String.prototype.concat
- String.prototype.constructor
- String.prototype.indexOf
- String.prototype.lastIndexOf
- String.prototype.localeCompare
- String.prototype.match
- String.prototype.replace
- String.prototype.search
- String.prototype.slice
- String.prototype.split
- String.prototype.substr
- String.prototype.substring
- String.prototype.toLocaleLowerCase
- String.prototype.toLocaleUpperCase
- String.prototype.toLowerCase
- String.prototype.toString
- String.prototype.toUpperCase
- String.prototype.trim
- String.prototype.valueOf
- (Objeto global) eval (x)
- (Objeto global) parseInt (string , radix)
- (Objeto global) parseFloat (string)
- (Objeto global) isNaN (number)
- (Objeto global) isFinite (number)



# #Objeto String

Utilizadas para manipulação de textos  
armazena as Strings como objetos do tipo String

```
1 var exemploString = 'Teste';
2 console.log(typeof exemploString);
3
4 var exemploString2 = new String('Teste');
5 console.log(typeof exemploString2);
6
```

```
>
string
object
>
```



# #Objeto String

```
1 //Propriedade
2 console.log('Senai'.length);
3 //Metodos
4 var senai = 'Olá o Senai!';
5 //retorna a posição da ocorrência
6 var n = senai.indexOf('Senai');
7 console.log('Achou na posição:',n);
8 //retorna a ultima ocorrência na cadeia
9 var ln = senai.lastIndexOf('ABC');
10 console.log('Achou na posição:',ln);
11 //Procura caracter na posição indicada
12 var sn = senai.charAt(2);
13 console.log('Achou:',sn);
14 //ao contrario: toLowerCase();
15 console.log('Maiusculo:',senai.toUpperCase());
16 //substituir
17 var rep = senai.replace('Senai','Cimatec');
18 console.log('Substituir:',rep);
19 //quebra string
20 console.log('Quebrei:',senai.split(' '));
21 //pegando pedaço da string
22 console.log('Pedaço:',senai.substring(1,4));
```

```
•
5
Achou na posição: 6
Achou na posição: -1
Achou: á
Maiusculo: OLÁ O SENAI!
Substituir: Olá o Cimatec!
Quebrei: [ 'Olá', 'o', 'Senai!' ]
Pedaço: lá
•
```



# #Objeto Array

- Trata-se de automatizar a declaração de um grande número de dados;
- As variáveis assim declaradas se acessam através de um índice.

```
1 //Declarando
2 var senaiList = ['Olá', 'Senai'];
3 console.log(senaiList);
4
5 var senaiObjArray = new Array('Olá', 'Senai');
6
7 senaiObjArray[3] = 'Cimatec';
8
9 console.log(senaiObjArray);
10
```

```
> [ 'Olá', 'Senai' ]
[ 'Olá', 'Senai', 'Cimatec' ]
>
```



# #Objeto Array

- forEach, map e filter ....

```
1 //Metodos
2 var senaiList = ['Olá', 'Senai','Cimatec'];
3 // varrendo array
4 var senaiList.forEach(function(obj,key){
5     console.log(key, obj);
6 });
7 // Mapeando e modificando
8 var list = senaiList.map(function(obj){
9     return (obj.indexOf('Senai') === 0)?'Yoda':obj;
10 });
11 console.log(list);
12 // Filtrando array
13 var filter = senaiList.filter(function(obj){
14     return (obj.indexOf('Senai') === 0)?true:false;
15 });
16 console.log(filter);
17 //transfoma array em string
18 console.log(senaiList.join(' '));
19 |
```

```
: 
0 'Olá'
1 'Senai'
2 'Cimatec'
[ 'Olá', 'Yoda', 'Cimatec' ]
[ 'Senai' ]
Olá Senai Cimatec
: |
```



# #Objeto Array

- Inserir, pegar e deletar

```
var senaiList = ['Olá', 'Senai','Cimatec'];

//Inserindo registro no Array
senaiList.push('Salvador');
senaiList.push('Bahia');
console.log('ex1:',senaiList);
//retorna array baseado na posição inicial e final -1
var nomeCidadeEstado = senaiList.slice(2,5);
console.log('ex2:',nomeCidadeEstado);
console.log('ex2:',senaiList); // array original ----
//removendo registro array [POSIÇÃO, ]
var itemRemovido = senaiList.splice(4,1);
console.log('ex3:',senaiList); // array original ----
console.log('ex3:',itemRemovido); // array original
----
```

```
•
ex1: [ 'Olá', 'Senai', 'Cimatec', 'Salvador', 'Bahia' ]
ex2: [ 'Cimatec', 'Salvador', 'Bahia' ]
ex2: [ 'Olá', 'Senai', 'Cimatec', 'Salvador', 'Bahia' ]
ex3: [ 'Olá', 'Senai', 'Cimatec', 'Salvador' ]
ex3: [ 'Bahia' ]
```



# #Objeto Date

- É um objeto que facilita a manipulação com horas e datas em JavaScript;
- Os valores do mês são contados de 0 até 11 e os dias da semana de 0 a 6 da seguinte forma

```
1 //Manipulando
2 //new Date(ano, mês, dia, hora, minuto, segundo, milissegundo);
3 console.log(Date.now(), '=> Data em milisegundos');
4 var hoje = new Date(Date.now());
5 console.log(hoje);
6 var dNoventa = new Date('1990-02-02');
7 console.log(dNoventa.toString());
8 var data = new Date('01/30/1990');
9 console.log(data.toString());
10 console.log(data.toDateString());
11
12 |
13
```

```
>
1429555945978 => Data em milisegundos
Mon, 20 Apr 2015 18:52:25 GMT
Thu Feb 01 1990 21:00:00 GMT-0300 (Hora Padrão
da Bahia)
Tue Jan 30 1990 00:00:00 GMT-0300 (Hora Padrão
da Bahia)
Tue Jan 30 1990
> |
```



# #Objeto Date

MÉTODOS	DESCRIÇÃO
getDate	Dia da semana (0=Domingo).
getDay	Dia do mês.
getHours	Horas (0 a 23).
getMinutes	Minutos (0 a 59).
getMonth	Mês do ano (0=janeiro).
getSeconds	Segundos (0 a 59).
getTime	Milissegundos desde 1 de janeiro de 1990 (00:00:00).
getTimezoneOffset	Diferença de fuso horário em minutos para o GMT.
getYear	2 dígitos do ano até 1999. Após 2000, 4 dígitos.

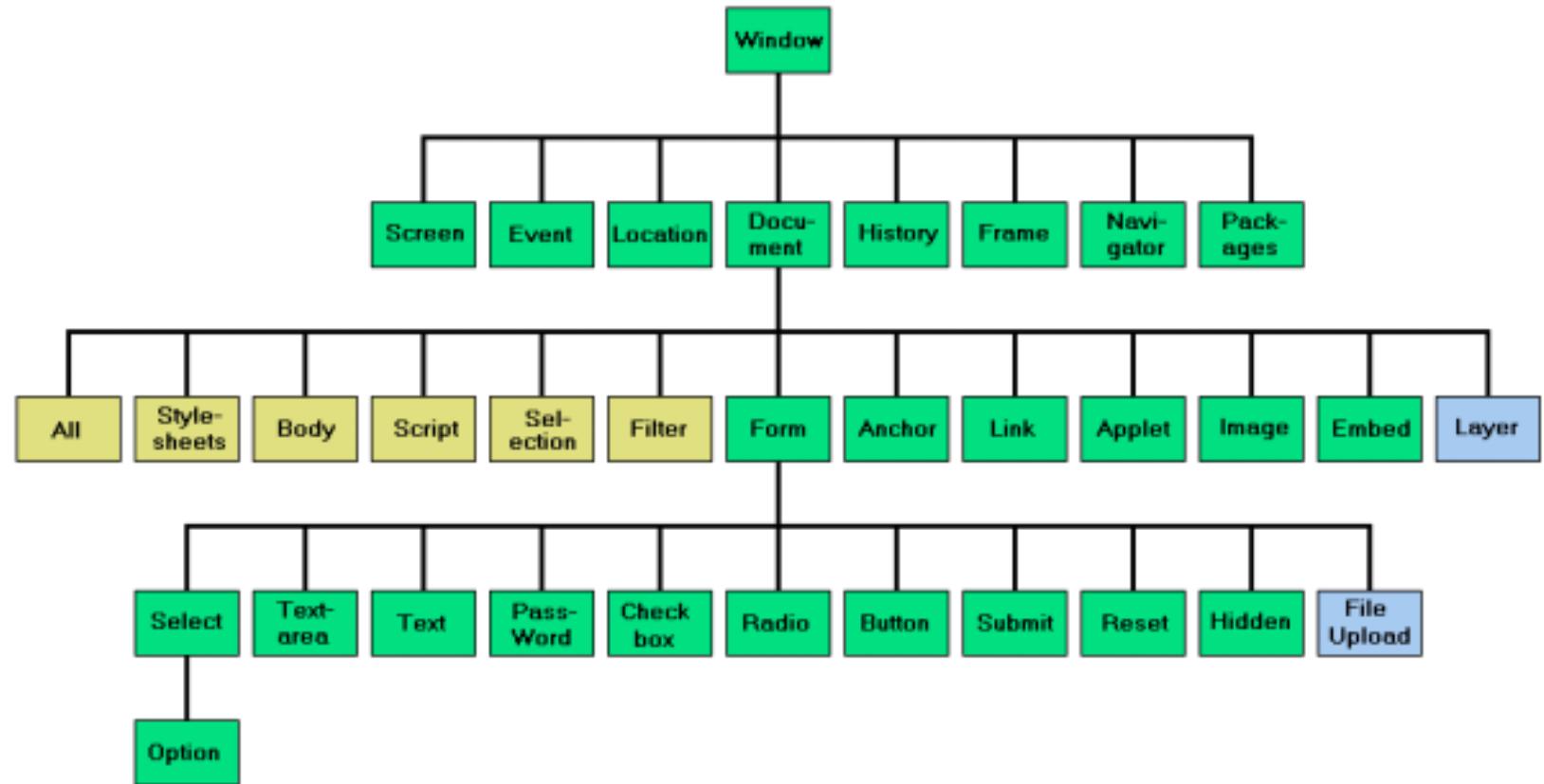


Window



# #Window

O objeto Window é o principal ponto de entrada para todos os recursos e APIs de Javascript do lado do cliente. Ele representa uma janela ou quadro de navegador Web e pode ser referenciado através do identificador window.



# #Window Propiedades

JS



closed	screenX
defaultStatus	screenY
document	scrollX
frameElement	scrollY
frames	self
history	status
innerHeight	top
innerWidth	
length	
location	
name	
navigator	
opener	
outerHeight	
outerWidth	
pageXOffset	
pageYOffset	
parent	
screen	
screenLeft	
screenTop	

# #Window Metodos

JS



alert()	resizeBy()
atob()	resizeTo()
blur()	scrollBy()
btoa()	scrollTo()
clearInterval()	setInterval()
clearTimeout()	setTimeout()
close()	stop()
confirm()	
createPopup()	
focus()	
moveBy()	
moveTo()	
open()	
print()	
prompt()	



# #Funções nativas do objeto window

Alert => Alerta no navegador, bloqueando ate pressionar OK

The screenshot shows a browser developer tools interface with the "Elements" tab selected. In the console, the command `window.alert('Ola senai');` is typed and executed. A modal alert dialog is displayed, showing the message "Ola senai" and an "OK" button. The browser's address bar shows "The page at https://www.google.com.br says:".

```
IAC Online JSON Viewer JS Bin - Collaborativ... Bem-vindo(a) de volta
Elements Network Sources Timeline Profiles Resources
<top frame> ▾ Preserve log
> window.alert('Ola senai');
>
```

The page at https://www.google.com.br says:

Ola senai

OK



# #Funções nativas do objeto window

Confirm => Retorna booleano para confirmação

The screenshot shows a browser's developer tools open, specifically the Console tab. The console log contains the following code:

```
<top frame> <input checked="" type="checkbox"/> Preserve log
> var resp = window.confirm('Você esta no Cimatec?');
```

To the right of the console, a JavaScript confirmation dialog is displayed. The dialog has a title bar labeled "JavaScript" and a message area containing the text "Você esta no Cimatec?". At the bottom are two buttons: "OK" and "Cancelar".



# #Funções nativas do objeto window

Prompt => Janela com Entrada de dados

The screenshot shows a browser's developer tools open, specifically the Console tab. The user has typed the following JavaScript code:

```
var resp = window.prompt('Digite aonde você esta:', 'Cimatec,Outros...');
```

When the code is run, a modal dialog box titled "JavaScript" appears. It contains the prompt message "Digite aonde você esta:" followed by a text input field containing the value "Cimatec,Outros...". At the bottom of the dialog are two buttons: "OK" and "Cancelar" (Cancel).



# #Funções nativas do objeto window

- `setTimeout("função",intervalo);`
  - Exemplo:
    - `var num = setTimeout(funcao,4000);`
- `clearTimeout();`
  - Exemplo:
    - `clearTimeout(num);`



# #Funções nativas do objeto window

- `setInterval("função",intervalo);`
  - Exemplo:
    - `var num = setInterval (funcao,4000);`
- `clearInterval();`
  - Exemplo:
    - `clearInterval(num);`



# #Funções nativas do objeto window

```
var senaiList = [];

function insertInInterval() {
  var listInterval = setInterval(function() {
    senaiList.push('I Love Js');
    console.log(senaiList);
    if(senaiList.length === 3){
      clearInterval(listInterval);
    }
  },100);
}
insertInInterval();
```

```
> [ 'I Love Js' ]
[ 'I Love Js', 'I Love Js' ]
[ 'I Love Js', 'I Love Js', 'I Love Js' ]
> █
```



# Document Object Model



# #DOM

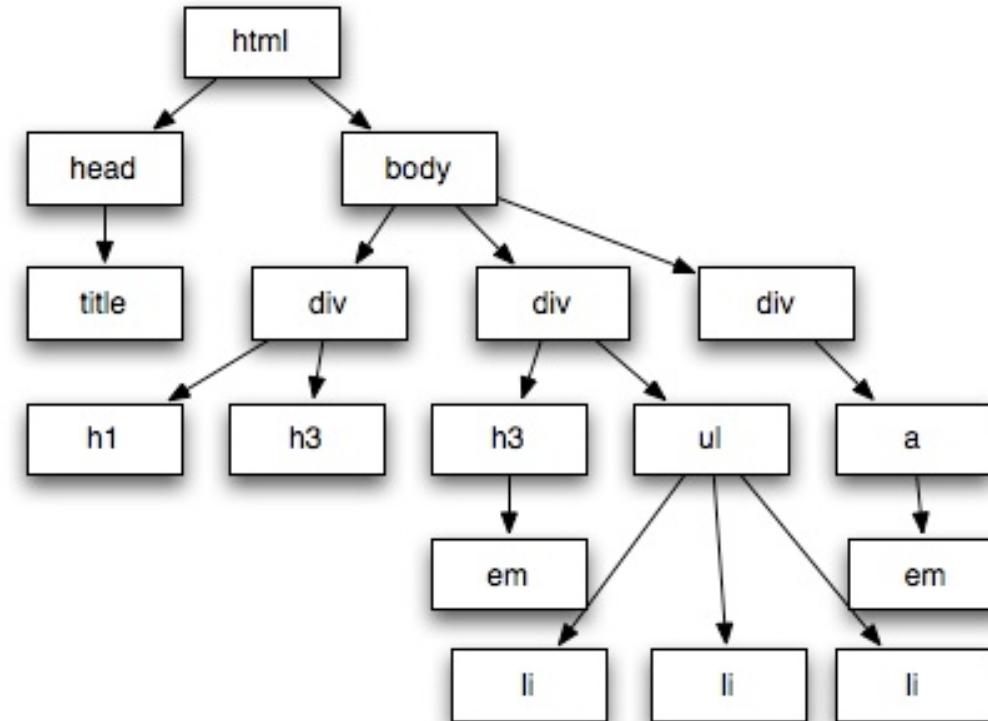
API(Conjunto de funções) que padroniza a estrutura de documentos HTML e XML, simplificando a tarefa de acessar e modificar tais documentos.

Se divide em: DOM HTML, DOM CORE e DOM XML

Segundo W3C<sup>1</sup>:

“O Modelo de Objetos do Documento (DOM, na sigla em inglês) é a interface entre a linguagem Javascript e os objetos do HTML. DOM é o método padrão para construção de aplicações ricas com Javascript e é amplamente conhecido e utilizado.”

1 - <http://www.w3c.br/cursos/html5/conteudo/>



# #Document Metodos

JS



document.activeElement

document.addEventListener()

document.adoptNode()

document.anchors

document.applets

document.baseURI

document.body

document.close()

document.cookie

document.createAttribute()

document.createComment()

document.createDocumentFragment()

document.createElement()

document.createTextNode()

document.doctype

document.inputEncoding

document.lastModified

document.links

document.normalize()

document.normalizeDocument()

document.open()

document.querySelector()

document.querySelectorAll()

document.doctype

document.documentElement

document.documentElementMode

document.documentElementURI

document.domain

document.domConfig

document.embeds

document.forms

document.getElementById()

document.getElementsByClassName()

document.getElementsByName()

document.getElementsByTagName()

document.hasFocus()

document.head

document.images

document.implementation

document.importNode()

document.readyState

document.referrer

document.removeEventListener()

document.renameNode()

document.scripts

document.strictErrorChecking

document.title

document.URL

document.write()

document.writeln()

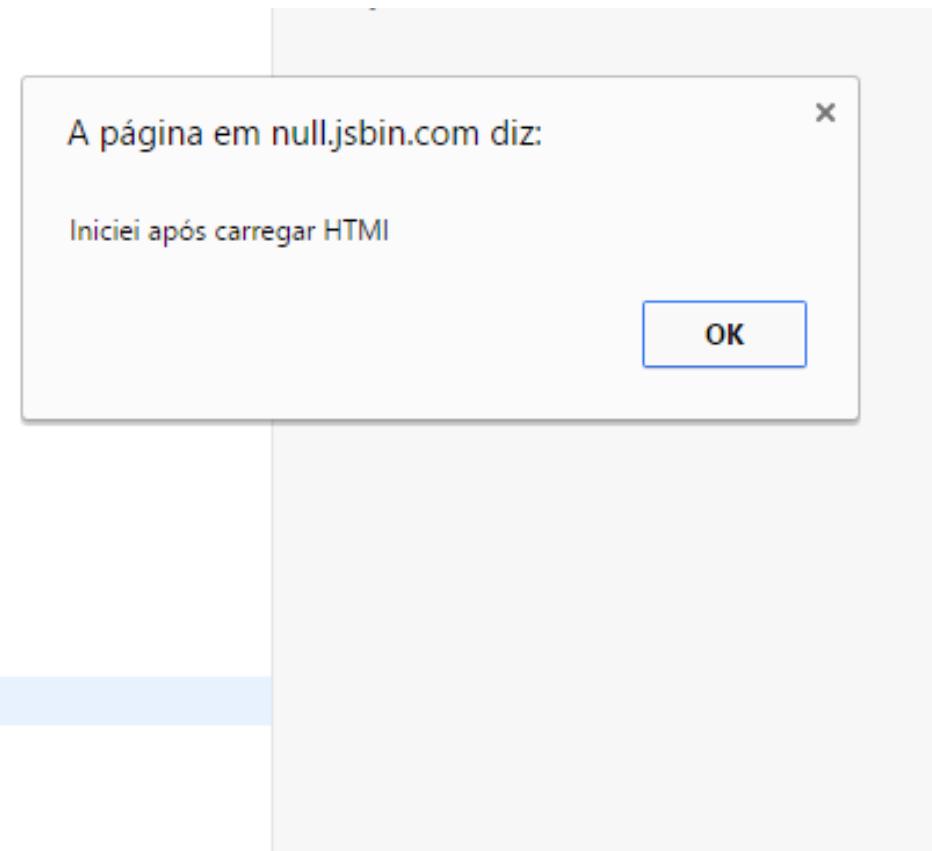


## #DOM

## Criando um função INIT

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>JS Bin</title>
  <script>
    window.onload = init;
    function init(){
      alert('Iniciei após carregar HTML');
    }
  </script>
</head>
<body>

</body>
</html>
```





## #DOM

getElementById = Pegando elemento pelo ID

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>JS Bin</title>
  <script>
    window.onload = init;
    function init(){
      var titulo = document.getElementById('titulo');
      titulo.innerHTML = 'Titulo da Pagina';
    }
  </script>
</head>
<body>
  <header>
    <h1 id='titulo'></h1>
  </header>
</body>
</html>
```

**Titulo da Pagina**



## #DOM

getElementsByTagName = Pegando elementos pela TAG

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>JS Bin</title>
  <script>
    window.onload = init;
    function init(){
      var titulo = document.getElementsByTagName('h1');
      titulo[0].innerHTML = 'Titulo da Pagina';
    }
  </script>
</head>
<body>
  <header>
    <h1></h1>
  </header>
</body>
</html>
```

# Titulo da Pagina



## #DOM

getElementsByName = Pegando elementos pelo atributo name

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>JS Bin</title>
  <script>
    window.onload = init;
    function init(){
      var nome = document.getElementsByName('nome');
      nome[0].value = 'Senai';
    }
  </script>
</head>
<body>
  <form action="">
    <input type="text" name="nome">
  </form>
</body>
</html>
```

A screenshot of a web browser window. Inside the window, there is a single text input field. The input field has a light gray border and contains the text "Senai" in a black sans-serif font. The rest of the page is blank white space.



## #DOM

getElementsByClassName = Pegando elementos pelo atributo class

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>JS Bin</title>
  <script>
    window.onload = init;
    function init(){
      var titulo = document.getElementsByClassName('header-title');
      titulo[0].innerHTML = 'Titulo da Pagina';
    }
  </script>
</head>
<body>
  <header>
    <h1 class="header-title"></h1>
  </header>
</body>
</html>
```

**Titulo da Pagina**





## #DOM

querySelector = Pegando elemento através de um query “.” Classe, “#” ID

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>JS Bin</title>
  <style>

  </style>
  <script>
    window.onload = init;
    function init(){
      var titulo = document.querySelector('.header-title');
      titulo.innerHTML = 'Titulo da Pagina';
    }
  </script>
</head>
<body>
  <header>
    <h1 class="header-title"></h1>
  </header>
</body>
</html>
```

**Titulo da Pagina**

HTML  
5

# #DOM

querySelectorAll = Pegando elementos através de uma query

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>JS Bin</title>
    <style>
        .zebraon{background:silver}
    </style>
    <script>
        window.onload = init;
        function init(){
            var zebra = document.querySelectorAll('.zebra tbody tr');
            var zebraLength = zebra.length;
            for(var i=0;i<zebraLength;i+=2){
                zebra[i].className='zebraon'
            }
        }
    </script>
</head>
<body>
    <table class="zebra">
        <thead>
            <tr>
                <th>Nome</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Senai</td>
            </tr>
            <tr>
                <td>Cimatec</td>
            </tr>
        </tbody>
    </table>
</body>
</html>
```

Nome  
Senai  
Cimatec

JS



## #DOM

Dataset = atribuir dados arbitrários a um elemento HTML qualquer, prefixando seus atributos com "data-".

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>JS Bin</title>
  <script>
    window.onload = init;
    function init(){
      var mac = document.querySelector('#mac'),
          caption = document.querySelector('.caption');
      caption.innerHTML = 'Processador: '+mac.dataset.processor;
    }
  </script>
</head>
<body>
  <figure>
    
    <figcaption class="caption"></figcaption>
  </figure>
</body>
</html>
```



Processador: Intel Core I7





## #DOM

Style = pegando ou inserindo css nos elementos

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>JS Bin</title>
  <script>
    window.onload = init;
    function init(){
      var titulo = document.querySelector('.header-title');
      titulo.innerHTML = 'Titulo da Pagina';
      titulo.style['background-color'] = 'green';
      titulo.style.color='#fff';
      titulo.style.fontSize='25px';
    }
  </script>
</head>
<body>
  <header>
    <h1 class="header-title"></h1>
  </header>
</body>
</html>
```

**Titulo da Pagina**



## #DOM

createElement = Criando um novo elemento

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>JS Bin</title>
<script>
  window.onload = init;
  function init(){
    var titulo = document.createElement('h1');
    titulo.innerHTML = 'Titulo da Pagina';
    document.body.appendChild(titulo);
  }
</script>
</head>
<body>
</body>
</html>
```

**Titulo da Pagina**

# #DOM - Eventos

JS



Window event att	Form event	Keyboard and Mouse event	Clipboard and Media event
onafterprint	onblur	onkeydown	oncopy
onbeforeprint	onchange	onkeypress	oncut
onbeforeunload	oncontextmenu	onkeyup	onpaste
onerror script	onfocus	onclick	onabort
onhashchange	oninput	ondblclick	onemptied
onload	oninvalid	ondrag	onended
onmessage	onreset	ondragend	onerror
onoffline	onsearch	ondragenter	onloadeddata
ononline	onselect	ondragleave	onloadedmetadata
onpagehide	onsubmit	ondragover	onloadstart
onpageshow		ondragstart	onpause
onpopstate		ondrop	onplay
onresize		onmousedown	onplaying
onstorage		onmousemove	onprogress
onunload		onmouseout	onratechange
		onmouseover	onseeked





# #DOM - Eventos

How to use:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Javascript</title>
</head>
<body>
  <button onclick="changeColor(this, 'red')>Clique</button>
  <script>
    function changeColor(element, color){
      element.style.color = color;
    }
  </script>
</body>
</html>
```

Clique



# #DOM - Eventos

How to use:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Javascript</title>
</head>
<body>
  <button id="btn">Clique</button>
  <script>
    var btn = document.querySelector('#btn');
    btn.onclick = function(){
      this.style.color = 'red';
    }
  </script>
</body>
</html>
```

Clique



# #DOM - Eventos

How to use:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Javascript</title>
</head>
<body>
  <button id="btn">Clique</button>
  <script>
    var btn = document.querySelector('#btn');
    btn.addEventListener('click', function(){
      this.style.color = 'red';
    }, false);
  </script>
</body>
</html>
```

Clique

Obs: Para o IE 8 ou inferior use `btn.attachEvent('onclick', function(){});`



JavaScript  
O.O.

The text "JavaScript O.O." is displayed in large, bold, yellow letters. It is positioned over a circular opening in a dark wooden door. Above the door is a circular emblem featuring a winged insect-like creature. The entire scene is set against a backdrop of a stone castle tower with a skull-like face integrated into its structure, all under a red sky.



# #JS OO Definição

“A linguagem Javascript suporta programação orientada a objetos (OOP). É mais apropriado dizer que Javascript é uma linguagem capaz de simular muitos dos fundamentos de OOP, embora não plenamente alinhada com todos os conceitos de orientação a objeto.”

(Maurício Samy – Javascript Guia do Programador, 2010 )



# #JS OO Classe

Javascript difere-se de linguagens clássicas orientadas a objeto como Java e C++ principalmente por não possuir uma definição formal de classe.

```
var Classe = function(){
};

function Classe2(param){
};

var c = new Classe(); //Objeto
```



# #JS OO Instância/Objeto

O operador unário new destina-se a criar uma instância de um objeto nativo ou de um objeto que tenha sido previamente definido.

Exemplo:

```
NomeObjeto = new construtor(args);
```

**NomeObjeto**: Nome escolhido do Objeto a Criar.

**construtor**: Expressão que constitue em uma função construtora, seguido de zero ou mais argumentos



# #JS OO Atributo e Método

A palavra-chave `this` deve ser usada em uma função ou método com a finalidade de criar uma referência para o objeto que define o método. Isso quer dizer que `this` deve estar contido no corpo da função.

Métodos em Javascript são funções invocadas por objetos.

```
var Carro = function(marca,modelo){  
    this.marca = marca;  
    this.modelo = modelo;  
  
    this.andar = function(){  
        //metodo  
    };  
};
```



# #JS OO Atributo e Método

## Acessando

Para acessar as propriedades ou métodos de um objeto você deve utilizar o operador “.” que deve ser precedido de uma referência ao objeto e sucedido pelo nome de uma de suas propriedades.

## Exemplo:

```
var carro = new Carro("Renault", "Logan");
carro.marca; // retorna Renault
```



# #JS OO Atributo e Método

Diferente das linguagens clássicas orientadas a objeto, Javascript permite que propriedades sejam adicionadas a qualquer momento durante a execução do código.

```
var carro = new Carro("Renault","Logan");

carro.cor = "Verde";
console.log(carro.cor); //Verde
```



# #JS OO Modificadores de acesso

O Javascript não é tipado, e não há modificadores de acesso em Javascript. Contudo conseguimos através da sintaxe da linguagem simular o que seria um método privado declarando uma variável, Exemplo:

```
var Pessoa = function(nome){  
    var nome = nome;  
  
    this.getNome = function(){  
        return nome.toString();  
    }  
}  
  
var p = new Pessoa("Carol");  
p.nome; //undefined  
p.getNome(); //Carol
```



# #JS OO Constantes

A implementação atual de const é uma extensão do Mozilla específica e não faz parte do ECMAScript 5. Ele é suportado no Firefox e Chrome (V8) e parcialmente suportada no Opera 9 + e Safari. Não é suportado no Internet Explorer 6-9, ou na visualização do Internet Explorer 10. A palavra-chave const atualmente declara a constante no escopo da função (como variáveis declaradas com var).

Exemplo:

```
const numero = 7;
```



# #JS OO Herança

Em Javascript a herança ocorre por meio de objetos protótipos, que é referenciado pela propriedade **prototype**.

```
function Eletrodomestico() {
    this.ligado = false;
    this.ligar = function() {
        this.ligado = true;
    }
    this.desligar = function() {
        this.ligado = false;
    }
}

function Ventilador(velMax) {
    var maximaPermitida = 5; // Uso de encapsulamento
    var velocidadePadrao = 3; // Variáveis privadas
    this.velocidadeMaxima = velMax;
}

Ventilador.prototype = new Eletrodomestico(); // Define o objeto protótipo
ventilador = new Ventilador(4);
alert(ventilador.ligado); // Retorna false
ventilador.ligar();
alert(ventilador.ligado); // Retorna true
```



# #JS OO Polimorfismo

Em Javascript a melhor forma de fazer Polimorfismo seria usando o método da Prototipagem , Exemplo:

```
function A(){
    this.x = 1;
}

A.prototype.DoIt = function(){
    return this.x += 1;
}

function B(){
    this.x = 1;
}

B.prototype.DoIt = function(){
    return this.x += 2;
}

a = new A;
b = new B;

a.DoIt(); //2
b.DoIt(); //3
```



# #JS OO Literais de Objeto

Os literais de objeto possibilitam criar e iniciar objetos de uma maneira diferente.

Sendo um tipo de dado constituído por uma coleção de dados, ou seja, é uma unidade que armazena dados formatados em pares [nome/valor](#), em definição formal, podemos dizer que é uma coleção não ordenada de propriedades e métodos constituída por pares nome/valor. Literais de Objetos foram a base para criação do JSON(Javascript Object Notation) que usam o mesmo formato com diferença que seus ‘nomes’ são escritos como string.



# #JS OO Literais de Objeto

```
var Carro = {  
    marca: "Renault",  
    modelo: "Logan",  
    dimensoes: {  
        largura: "1.735mm",  
        altura: "1.525mm",  
        comprimento: "4.250mm"  
    },  
    andar: function(){  
        //metodo andar  
    }  
};  
  
console.log(Carro.marca); //Renault
```



# #JS OO Literais de Objeto

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Generic JSON</title>
</head>
<body>
  <script type="text/javascript">
    var p1 = {"nome":"pedro", "sobrenome":"abs", "idade":35};
    var p2 = {nome:"pedro", sobrenome:"abs", idade:35}

    window.onload = function(){
      document.write(p1.idade + "<br>");
      document.write(p2.idade + "<br>");
      document.write(typeof p1.idade + "<br>");
      document.write(typeof p2.idade + "<br>");
    }
  </script>
</body>
</html>
```

```
35
35
number
number
```

JS



# #Atividade





## #Atividade:

1. Pegue a função assert criado na aula anterior, insira mais um parâmetro aonde receberá a descrição do teste e imprima em forma de lista no html. Crie duas classes no css “.pass” para se o teste passar com a cor verde e “.fail” para se o teste não passar com a cor vermelha
2. Crie um relógio e faça se atualizar a cada 100 milissegundos. Use o método setInterval
3. Crie uma lista de tarefas com ul e li, e quando eu clicar em cima da tarefa ele mude a cor para vermelho e coloque uma linha em cima do texto



# #Atividade assert

```
var somar = function(a,b){  
    return a+b;  
};  
  
function assert(resultado, fn){  
    (resultado === fn)? console.log('Teste Passou'): console.log('Teste Falhou');  
}  
  
assert(4, somar(2,2));
```



# #Resposta atividade 1

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>JS Bin</title>
    <style type="text/css">
        #results li.pass {color: green;}
        #results li.fail {color: red;}
    </style>
</head>
<body>
    <ul id="results"></ul>
    <script>
        var soma = function(a,b){
            return a+b;
        };

        function assert(resultado, fn, desc){
            var results = document.getElementById("results");
            var li = document.createElement("li");
            li.className = (resultado === fn) ? "pass" : "fail";
            li.appendChild(document.createTextNode(desc));
            results.appendChild(li);
        }

        assert(4, soma(2,2), 'somando variaveis');
        assert(6, soma(2,2), 'somando variaveis');
    </script>
</body>
</html>
```

- somando variaveis
- somando variaveis



# #Resposta atividade 2

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>JS Bin</title>
</head>
<body>
    <div class="relogio">
        <span id="hora"></span>:
        <span id="minuto"></span>:
        <span id="segundo"></span>
    </div>

    <script>
        function getHour(){
            var data = new Date();
            document.querySelector('#hora').innerHTML = data.getHours();
            document.querySelector('#minuto').innerHTML = data.getMinutes();
            document.querySelector('#segundo').innerHTML = data.getSeconds();
            getHour();
        }
        window.setInterval(getHour,100);
    </script>
</body>
</html>
```



# #Resposta atividade 3

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Javascript</title>
    <style>
        .tarefas li{ cursor:pointer; }
    </style>
</head>
<body>
    <ul class="tarefas">
        <li>Tarefa 01</li>
        <li>Tarefa 02</li>
    </ul>
    <script>
        var tarefas = document.querySelectorAll('.tarefas li');
        for(var i=0; i<tarefas.length;i++){
            tarefas[i].addEventListener('click',function(){
                this.style.color = 'red';
                this.style.textDecoration = 'line-through';
            }, false);
        }

    </script>
</body>
</html>
```

- Tarefa 01
- Tarefa 02